

**AMENDMENTS TO THE CLAIMS:**

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (currently amended) A data processing apparatus comprising:

arbitration logic; and

a data processor core, said data processor core comprising:

a memory access interface portion for performing data transfer operations between an external data source and at least one memory associated with said data processor core;

a data processing portion for performing data processing operations;

a read/write port for ~~transferring~~ transferring data from said processor core to ~~at least two~~ first and second buses, said ~~at least two~~ first and second buses providing data communication between said processor core and said at least one memory, said at least one memory comprising ~~at least two~~ first and second memory portions, ~~each of said at least two buses~~ said first bus providing exclusive data access to ~~respective ones of said at least two portions~~ said first memory portion and said second bus providing exclusive access to said second memory portion, wherein said arbitration logic is associated with said read/write port and said arbitration logic routes a data access request requesting access of data in said first memory ~~one portion of said at least one memory~~ received from said memory access interface to ~~one of said at least two buses~~ said first bus ~~providing access to said one portion of said at least one memory~~ and routes a further data access request requesting access of data in a further said second memory ~~portion of said at least one memory~~ received from said data processing portion to a further ~~one of said at least two~~

~~buses~~said second bus ~~providing access to said further portion of said at least one memory,~~ said routing of said data access requests being performed during the same clock cycle.

2. (currently amended) A data processing apparatus according to claim 1, said arbitration logic selecting one of said ~~at least two~~ first and second buses to which said data access request is routed in dependence upon an address location within said at least one memory associated with said data access request.

3. (currently amended) A data processing apparatus according to claim 2, wherein said ~~at least two~~ first and second portions of said memory comprise an instruction portion storing instructions and at least one data portion storing data items, respectively, said arbitration logic routing said data access request to a said ~~first one of said at least two buses~~ bus providing access to said instruction portion when data to be transferred is an instruction and routing said data access request to a said ~~second of said at least two buses~~ bus providing access to said at least one data portion when data to be transferred is a data item.

4. (currently amended) A data processing apparatus according to claim 3, wherein said at least one data portion comprises two data portions, an even data portion storing data having an even address and an odd data portion storing data having an odd address, said read/write port transferring data between said processor core and said at least one memory via three buses, a said first bus providing access to said instruction portion, a said second bus providing access to said odd data portion, and a third bus providing access to said even data portion, and said arbitration logic routing a data access request to said first bus when data to be transferred is an instruction,

to said second bus when data to be transferred is a data item associated with an odd address and to said third bus when data to be transferred is a data item associated with an even address.

5. (currently amended) A data processing apparatus according to claim 1, wherein said arbitration logic, in response to receipt of a data access request from said memory access interface portion and a data access request from said data processing portion, both data access requests requesting access to data in one portion of said at least one memory, routing said data access request from said memory access interface portion to one of said first and second ~~at least two~~ buses providing data access to said one portion of said at least one memory before routing said request from said processing portion to said one of said ~~at least two~~ first and second buses.

6. (currently amended) A data processing apparatus according to claim 1, said arbitration logic detecting a wait request from at least one busy portion of said at least one memory, said arbitration logic not routing any data access requests to said busy portion until said wait request is no longer detected.

7. Canceled.

8. (currently amended) A data processing apparatus according to claim ~~7~~1, wherein said at least one memory is divided into three portions, an instruction portion storing instructions, and two data portions, an even data portion storing data having an even address, and an odd data portion storing data having an odd address, said data processing apparatus comprising three buses, said read/write port transferring data between said processor core and said at least one

memory via said three buses, a said first bus providing access to said instruction portion, a said second bus providing access to said odd data portion, and a third bus providing access to said even data portion.

9. (original) A data processing apparatus according to claim 7, wherein said at least one memory is a tightly coupled memory.

10. (currently amended) A method of transferring data between an external data source and at least one memory associated with a data processor core, said data processor core comprising a memory access interface portion performing data transfer operations between said external data source and said at least one memory associated with said data processor core and a data processing portion performing data processing operations, said method comprising the steps of:

in response to a data access request requesting access of data in ~~one~~ a first memory portion of said at least one memory received from said memory access interface portion and a data access request requesting access to data in a ~~further~~ second memory portion of said at least one memory received from said data processing portion, routing said data access request received from said memory access interface portion to one of ~~at least two~~ first and second buses, said ~~one of said at least two buses~~ first bus providing exclusive access to said ~~one~~ first memory portion ~~of said at least one memory~~, and routing said data access request received from said data processing portion to a ~~further~~ second bus, said ~~further~~ second bus providing exclusive access to said ~~further~~ second memory portion ~~of said at least one memory~~, said routing of said data access requests being performed during the same clock cycle.

11. (original) A method according to claim 10, wherein said step of routing data access requests to respective data buses is done in dependence upon an address location within said at least one memory associated with said data access request.

12. (currently amended) A method according to claim 10, wherein said ~~at least two~~ first and second portions of said memory comprise an instruction portion storing instructions and at least one data portion storing data items, said step of routing said data access requests routing a data access request to ~~one of said at least two buses~~ said first bus providing exclusive access to said instruction portion when data to be transferred is an instruction and routing said data access request to ~~another of said at least two buses~~ said second bus providing exclusive access to said at least one data portion when data to be transferred is a data item.

13. (currently amended) A method according to claim 12, wherein said at least one data portion comprises two data portions, an even data portion storing data having an even address and an odd data portion storing data having an odd address, said routing step routing data accesses to one of three buses, a said first bus providing access to said instruction portion, a said second bus providing access to said odd data portion, and a third bus providing access to said even data portion, and said routing step routing a data access request to said first bus when data to be transferred is an instruction, to said second bus when data to be transferred is a data item associated with an odd address, and to said third bus when data to be transferred is a data item associated with an even address.

14. (original) A method according to claim 10, wherein said routing step in response to receipt of a data access request from said memory access interface portion and a data access request from said data processing portion, both data access requests requesting access to data in a portion of said at least one memory accessed by one of ~~said at least two~~ first and second buses, routes said data access request from said memory access interface portion to said one of said at ~~least two~~ first and second buses before routing said request from said processing portion to said one of said at ~~least two~~ first and second buses.

15. (original) A method according to claim 10, said routing step detecting a wait request from at least one busy portion of said at least one memory, and in response to detection of said wait request not routing any data access requests to said busy portion until said wait request is no longer detected.

16. (currently amended) A method according to claim 10, wherein said at least one memory is divided into three portions, an instruction portion storing instructions, and two data portions, an even data portion storing data having an even address and an odd data portion storing data having an odd address, said routing step routing a received data access request to one of three buses, said first bus providing access to said instruction portion, said second bus providing access to said odd data portion, and a third bus providing access to said even data portion, in dependence upon an address of said data associated with said data access request.

17. (previously presented) Arbitration logic controlling a data processor to perform the steps of the method according to claim 10.

18. (previously presented) A data processing apparatus according to claim 1, wherein said data processor core comprises said arbitration logic.

Claims 1-3, 5-7, 9-12, 14-15, and 17-18 stand rejected under 35 U.S.C. 102(b) for anticipation by US patent 4,979,100 to Makris. This rejection is respectfully traversed.

To establish that a claim is anticipated, the Examiner must point out where each and every limitation in the claim is found in a single prior art reference. *Scripps Clinic & Research Found. v. Genentec, Inc.*, 927 F.2d 1565 (Fed. Cir. 1991). Every limitation contained in the claims must be present in the reference, and if even one limitation is missing from the reference, then it does not anticipate the claim. *Kloster Speedsteel AB v. Crucible, Inc.*, 793 F.2d 1565 (Fed. Cir. 1986). Makris fails to satisfy this rigorous standard.

In some circumstances, it is particularly important that data can be transferred quickly to or from a particular memory. The application describes a processor core (e.g., 10 in Figure 1) that is divided into portions including a data processing portion (e.g., 12 in Figure 1) and a memory access portion (e.g., 30 in Figure 1). The division of the processor core into portions has several advantages. One is that data may be transferred faster than would be the case if the data was transferred via a standard processor core because the instructions required to read the data, store it in a register on the core, and then write it to a memory are not required.

But a problem associated with providing a memory access interface portion of the core along with the processing portion is that both portions may wish to access data stored in an associated memory at the same time. This problem can be particularly acute in applications such as data logging, where a large amount of data that does not need to be processed immediately is transferred via the core to a memory.

This problem is addressed by providing at least two buses (e.g., 75A and 75B in Figure 5) operable to provide data communication between the processor core and the associated memory.

The associated memory is divided into at least two portions, (e.g., 120A and 120B in Figure 5), and the two buses each provide exclusive access to their respective memory portion. Arbitration logic (e.g., 110 in Figure 5) routes data access requests to a particular portion of the memory to the appropriate corresponding bus. Having more than one bus allows data access requests received from the memory access interface and the data processing portion of the core in the same clock cycle to be processed in parallel along respective exclusive buses. This alleviates some of the problems that can occur when two portions of the core are trying to access one memory via a single bus at the same time.

In an example non-limiting embodiment, the arbitration logic selects one of the two buses to route the data access request depending on an address location within the memory associated with the data access request (e.g., even or odd address). The address associated with the data provides an indication of the particular portion of the memory where it is stored and is thus a useful indicator that the arbitration logic uses to determine which exclusive bus to use.

Makris relates to a packet switch which receives data and processes it for assembly into packages. Plural data processing units 62 and a bus 60 allows communication between each of the data processing units of the switch. One or more storage units 58 store the data packets. Because of the large number of processing units involved, a complicated arbitration scheme is needed to decide which of the processing units will be granted access to the bus. For example, Makris's arbitrator 59 selectively and alterably designates any of at least two different levels of priority of access to the bus for each of the processing units. The arbitration logic identified by the Examiner in Makris at column 18, lines 25-26 is within a PAL (programmable array logic) which is a physically separate device from the processing units. The PAL is complex logic

operable to arbitrate between twenty potential bus requests by grouping them into five subgroups and using a token system.

Makris does not disclose the “a read/write port for transferring data from said processor core to first and second buses...said first bus providing exclusive access to said first memory portion and said second bus providing exclusive access to said second memory portion,” as recited in claim 1. The Examiner relies on Makris’s two buses A and B, each of which a master can select to connect to any one of a plurality of slaves. “[A]ll slaves capture the information transmitted on the bus.” See col, 20, line 9. Thus, Markis fails to disclose the claimed first and second buses **each** providing **exclusive access** to its **respective** memory portion. The method claim 10 also recites routing a data access request to first and second memory portions via respective first and second buses that each provides exclusive access to its respective portion of the memory.

In Makris, multiple buses receive signals from masters and from peripherals, and various encodings are used in the data to enable the bus to know which device the data came from. Makris’s PAL arbitration logic arbitrates between twenty potential bus requests by grouping them into five subgroups and using a token system. Although the PAL arbiter generally looks to arbitrate between many devices on a single bus, in the case of more than one bus, the additional bus is not dedicated to a particular memory portion. Instead, the additional bus simply provides another alternative access to many of the slaves. Requests must specify the slave concerned, and all slaves must listen and wait to see if the request is for them. See Markis’s Abstract. Consequently, Markis does not teach using multiple dedicated buses where each bus provides exclusive access to a corresponding memory portion.